2. MATLAB Basics

2.1 *(a)* The size of `array1` is 4 × 5. *(b)* The value of `array1(4,1)` is 1.1. *(c)* `array1(:,1:2:5)` is a 4 × 3 array consisting of the first, third, and fifth columns of `array1`:

```
>> array1(:,1:2:5)
ans =
         0    2.1000    6.0000
         0   -6.6000    3.4000
    2.1000    0.3000    1.3000
    1.1000         0   -2.0000
```

(d) `array1([1 3],end)` consists of the elements in the first and third rows on the last column of `array1`:

```
» array1([1 3],end)
ans =
    6.0000
    1.3000
```

2.2 *(a)* Legal *(b)* Illegal—names must begin with a letter. *(c)* Legal *(d)* Illegal—names must begin with a letter. *(e)* Illegal—the apostrophe and the question mark are illegal characters.

2.3 *(a)* This is a three element row array containing the values 1, 3, and 5:

```
» a = 2:3:8
a =
     2     5     8
```

(b) This is a 3 × 3 element array containing three identical columns:

```
» b = [a' a' a']
b =
     2     2     2
     5     5     5
     8     8     8
```

(c) This is a 2 × 2 element array containing the first and third rows and columns of `b`:

```
» c = b(1:2:3,1:2:3)
c =
     2     2
     8     8
```

(d) This is a 1 × 3 row array containing the sum of `a` (= [2 5 8]) plus the second row of `b` (= [5 5 5]):

```
» d = a + b(2,:)
d =
```

```
     7    10    13
```

(e) This is a 1 × 9 row array containing:

```
» w = [zeros(1,3) ones(3,1)' 3:5']
w =
     0     0     0     1     1     1     3     4     5
```

Note that the expression `3:5'` is the same as `3:5`, because the transpose operator applies to the single element 5 only: `5' = 5`. Both expressions produce the row array `[1 3 5]`. To produce a column array, we would write the expression as `(3:5)'`, so that the transpose operator applied to the entire vector.

(f) This statement swaps *the first and third rows in the second column* of array `b`:

```
» b([1 3],2) = b([3 1],2)
b =
     2     8     2
     5     5     5
     8     2     8
```

(g) This statement produces nothing, because even the first element (1) is below the termination condition (5) when counting down:

```
» e = 1:-1:5
e =
   Empty matrix: 1-by-0
```

2.4 *(a)* This is the third row of the array:

```
» array1(3,:)
ans =
    2.1000    0.3000    0.1000   -0.4000    1.3000
```

(b) This is the third column of the array:

```
» array1(:,3)
ans =
    2.1000
   -5.6000
    0.1000
         0
```

(c) This array consists of the first and third rows and the third and fourth columns of `array1`, with the third column *repeated twice*:

```
» array1(1:2:3,[3 3 4])
ans =
   -2.1000   -2.1000   -3.5000
    0.1000    0.1000   -0.4000
```

(d) This array consists of the first row *repeated twice*:

```
» array1([1 1],:)
ans =
    1.1000         0   -2.1000   -3.5000    6.0000
    1.1000         0   -2.1000   -3.5000    6.0000
```

2.5 *(a)* This statement displays the number using the normal MATLAB format:

```
» disp (['value = ' num2str(value)]);
value = 31.4159
```

(b) This statement displays the number as an integer:

```
» disp (['value = ' int2str(value)]);
value = 31
```

(c) This statement displays the number in exponential format:

```
» fprintf('value = %e\n',value);
value = 3.141593e+001
```

(d) This statement displays the number in floating-point format:

```
» fprintf('value = %f\n',value);
value = 31.415927
```

(e) This statement displays the number in general format, which uses an exponential form if the number is too large or too small.

```
» fprintf('value = %g\n',value);
value = 31.4159
```

(f) This statement displays the number in floating-point format in a 12-character field, with 4 digits after the decimal point:

```
» fprintf('value = %12.4f\n',value);
value =      31.4159
```

2.6 The results of each case are shown below.

(a) Legal: This is element-by-element addition.

```
» result = a + b
result =
     1     4
    -1     6
```

(b) Legal: This is matrix multiplication. Since `eye(2)` is the 2 × 2 identity matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, the result of the multiplication is just the original matrix `a`.

```
» result = a * d
result =
     2     1
```

```
    -1     4
```

(c) Legal: This is element by element array multiplication

```
» result = a .* d
result =
     2     0
     0     4
```

(d) Legal: This is matrix multiplication

```
» result = a * c
result =
     5
     2
```

(e) Illegal: This is element by element array multiplication, and the two arrays have different sizes.

(f) Legal: This is matrix left division

```
» result = a \ b
result =
   -0.4444    1.1111
   -0.1111    0.7778
```

(g) Legal: This is element by element array left division: `b(i) / a(i)`

```
» result = a .\ b
result =
   -0.5000    3.0000
         0    0.5000
```

(h) Legal: This is element by element exponentiation

```
» result = a .^ b
result =
    0.5000    1.0000
    1.0000   16.0000
```

2.7 *(a)* 8.2 *(b)* 8.2 *(c)* 1.0 *(d)* 729 *(e)* 6561 *(f)* 729 *(g)* 4 *(h)* 4 *(i)* 3

2.8 *(a)* $0.0 + 25.0i$ *(b)* $-0.6224i$

2.9 The results of these expressions are:

(a)

```
>> exp(-2*t) * cos(omega * t)
ans =
    0.0183
```

(b)

```
>> exp(-2*t) * (cos(omega * t) + i*sin(omega * t))
ans =
   0.0183 - 0.0000i
```

(c)

```
>> exp(-2*t + i*omega*t)
ans =
   0.0183 - 0.0000i
```

Note that the answers in part *(b)* and part *(c)* are identical.

2.10 The solution to this set of equations can be found using the left division operator:

```
» a = [ -2.0 +5.0 +1.0 +3.0 +4.0 -1.0; ...
         2.0 -1.0 -5.0 -2.0 +6.0 +4.0; ...
        -1.0 +6.0 -4.0 -5.0 +3.0 -1.0; ...
         4.0 +3.0 -6.0 -5.0 -2.0 -2.0; ...
        -3.0 +6.0 +4.0 +2.0 -6.0 +4.0; ...
         2.0 +4.0 +4.0 +4.0 +5.0 -4.0 ];

» b = [ 0.0; 1.0; -6.0; 10.0; -6.0; -2.0];
» a\b
ans =
    0.6626
   -0.1326
   -3.0137
    2.8355
   -1.0852
   -0.8360
```

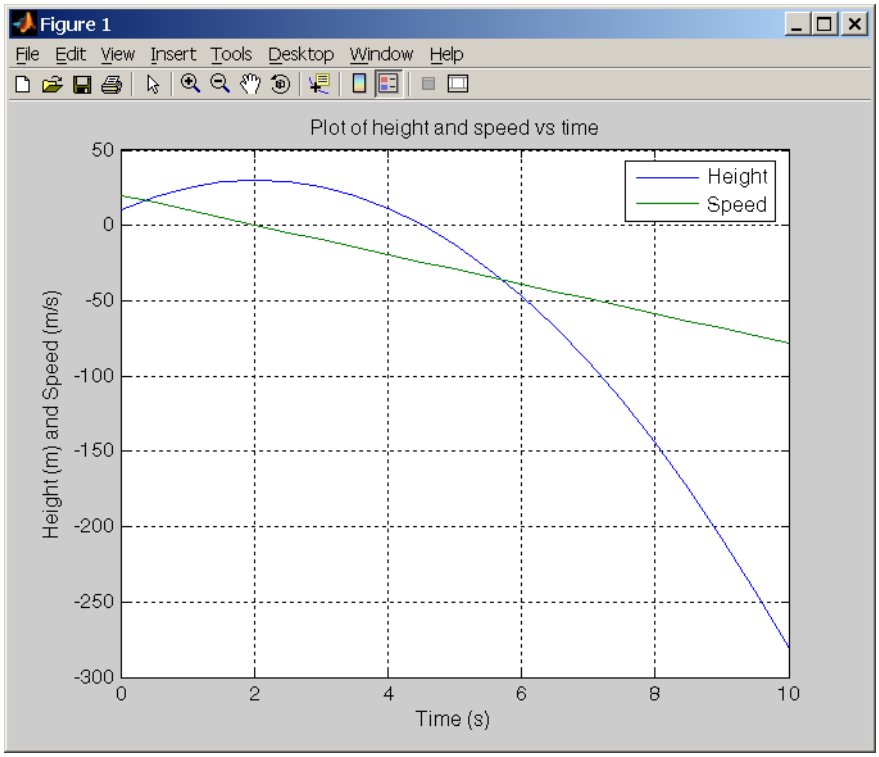
2.11 A program to plot the height and speed of a ball thrown vertically upward is shown below:

```
%  Script file: ball.m
%
%  Purpose:
%    To calculate and display the trajectory of a ball
%    thrown upward at a user-specified height and speed.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   g         -- Acceleration due to gravity (m/s^2)
%   h         -- Height (m)
%   h0        -- Initial height (m)
%   t         -- Time (s)
%   v         -- Vertical Speed (m/s)
%   v0        -- Initial Vertical Speed (m/s)

% Initialize the acceleration due to gravity
g = -9.81;
```

```
% Prompt the user for the initial velocity.
v0 = input('Enter the initial velocity of the ball: ');

% Prompt the user for the initial height
h0 = input('Enter the initial height of the ball: ');

% We will calculate the speed and height for the first
% 10 seconds of flight.  (Note that this program can be
% refined further once we learn how to use loops in a
% later chapter.  For now, we don't know how to detect
% the point where the ball passes through the ground
% at height = 0.)
t = 0:0.5:10;
h = zeros(size(t));
v = zeros(size(t));
h = 0.5 * g * t .^2 + v0 .* t + h0;
v = g .* t + v0;

% Display the result
plot(t,h,t,v);
title('Plot of height and speed vs time');
xlabel('Time (s)');
ylabel('Height (m) and Speed (m/s)');
legend('Height','Speed');
grid on;
```

When this program is executed, the results are:

```
» ball
Enter the initial velocity of the ball: 20
Enter the initial height of the ball: 10
```



2.12 A program to calculate the distance between two points in a Cartesian plane is shown below:

```
%  Script file: dist2d.m
%
%  Purpose:
%    To calculate the distance between two points on a
%    Cartesian plane.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   dist      -- Distance between points
%   x1, y1    -- Point 1
%   x2, y2    -- Point 2

% Prompt the user for the input points
x1 = input('Enter x1: ');
y1 = input('Enter y1: ');
x2 = input('Enter x2: ');
y2 = input('Enter y2: ');

% Calculate distance
dist = sqrt((x1-x2)^2 + (y1-y2)^2);

% Tell user
disp (['The distance is ' num2str(dist)]);
```

When this program is executed, the results are:

```
» dist2d
Enter x1: -3
Enter y1: 2
Enter x2: 6
Enter y2: -6
The distance is 10
```

2.13 *(a)* A program that accepts a 2D vector in rectangular coordinates and calculates the vector in polar coordinates is given below;

```
%  Script file: rect2polar.m
%
%  Purpose:
%    To calculate the polar coordinates of a vector given the
%    rectangular coordinates.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   x, y      -- Rectangular coordinates of vector
%   r         -- Length of vector
%   theta     -- Direction of vector in degrees

% Prompt the user for the input points
x = input('Enter x: ');
y = input('Enter y: ');

% Calculate polar coordinates.  Note that "180/pi" converts
% from radians to degrees.
r = sqrt(x^2 + y^2);
theta = atan2(y,x) * 180/pi;

% Tell user
disp (['The polar coordinates are ' num2str(r) ' at an angle of ' ...
num2str(theta) ' deg.']);
```

When this program is executed, the results are:

```
>> rect2polar
Enter x: 3
Enter y: 4
The polar coordinates are 5 at an angle of 53.1301 deg.
```

(b) A program that accepts a 2D vector in polar coordinates and calculates the vector in rectangular coordinates is given below;

```
%  Script file: polar2rect.m
```

```
%
%  Purpose:
%    To calculate the rectangular coordinates of a vector given
%    the polar coordinates.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   x, y      -- Rectangular coordinates of vector
%   r         -- Length of vector
%   theta     -- Direction of vector in degrees

% Prompt the user for the input points
r = input('Enter vector length r: ');
theta = input('Enter angle theta in degrees: ');

% Calculate polar coordinates.  Note that "pi/180" converts
% from radians to degrees.
x = r * cos(theta * pi/180);
y = r * sin(theta * pi/180);

% Tell user
disp (['The rectangular coordinates are x = ' num2str(x) ' and y = '  ...
num2str(y)]);
```

When this program is executed, the results are:

```
>> polar2rect
Enter vector length r: 5
Enter angle theta in degrees: 36.87
The rectangular coordinates are x = 3 and y = 4
```

2.14 A program to calculate the distance between two points in a three-dimensional Cartesian space is shown below:

```
%  Script file: dist3d.m
%
%  Purpose:
%    To calculate the distance between two points on a
%    Cartesian plane.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   dist       -- Distance between points
%   x1, y1, z1 -- Point 1
%   x2, y2, z2 -- Point 2
```

```
% Prompt the user for the input points
x1 = input('Enter x1: ');
y1 = input('Enter y1: ');
z1 = input('Enter z1: ');
x2 = input('Enter x2: ');
y2 = input('Enter y2: ');
z2 = input('Enter z2: ');

% Calculate dBm
dist = sqrt((x1-x2)^2 + (y1-y2)^2 + (z1-z2)^2);

% Tell user
disp (['The distance is ' num2str(dist)]);
```

When this program is executed, the results are:

```
» dist3d
Enter x1: -3
Enter y1: 2
Enter z1: 5
Enter x2: 3
Enter y2: -6
Enter z2: -5
The distance is 14.1421
```

2.15 *(a)* A program that accepts a 3D vector in rectangular coordinates and calculates the vector in spherical coordinates is shown below:

```
%  Script file: rect2spherical.m
%
%  Purpose:
%    To calculate the spherical coordinates of a vector given
%    the 3D rectangular coordinates.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   x, y, z   -- Rectangular coordinates of vector
%   r         -- Length of vector
%   theta     -- Direction of vector (x,y) plane, in degrees
%   phi       -- Elevation angle of vector, in degrees

% Prompt the user for the input points
x = input('Enter x: ');
y = input('Enter y: ');
z = input('Enter z: ');

% Calculate polar coordinates.  Note that "180/pi" converts
% from radians to degrees.
r = sqrt(x^2 + y^2 + z^2);
theta = atan2(y,x) * 180/pi;
```

```
phi = atan2(z,sqrt(x^2 + y^2)) * 180/pi;

% Tell user
disp ('The spherical coordinates are:');
disp (['r     = ' num2str(r)]);
disp (['theta = ' num2str(theta)]);
disp (['phi   = ' num2str(phi)]);
```

When this program is executed, the results are:

```
>> rect2spherical
Enter x: 4
Enter y: 3
Enter z: 0
The spherical coordinates are:
r     = 5
theta = 36.8699
phi   = 0

>> rect2spherical
Enter x: 4
Enter y: 0
Enter z: 3
The spherical coordinates are:
r     = 5
theta = 0
phi   = 36.8699
```

(b) A program that accepts a 3D vector in spherical coordinates (with the angles θ and ϕ in degrees) and calculates the vector in rectangular coordinates is shown below:

```
%  Script file: spherical2rect.m
%
%  Purpose:
%    To calculate the 3D rectangular coordinates of a vector
%    given the spherical coordinates.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   x, y, z   -- Rectangular coordinates of vector
%   r         -- Length of vector
%   theta     -- Direction of vector (x,y) plane, in degrees
%   phi       -- Elevation angle of vector, in degrees

% Prompt the user for the input points
r = input('Enter vector length r: ');
theta = input('Enter plan view angle theta in degrees: ');
phi = input('Enter elevation angle phi in degrees: ');
```

```
% Calculate spherical coordinates.  Note that "pi/180" converts
% from radians to degrees.
x = r * cos(phi * pi/180) * cos(theta * pi/180);
y = r * cos(phi * pi/180) * sin(theta * pi/180);
z = r * sin(phi * pi/180);

% Tell user
disp ('The 3D rectangular coordinates are:');
disp (['x = ' num2str(x)]);
disp (['y = ' num2str(y)]);
disp (['z = ' num2str(z)]);
```

When this program is executed, the results are:

```
>> spherical2rect
Enter vector length r: 5
Enter paln view angle theta in degrees: 36.87
Enter elevation angle phi in degrees: 0
The rectangular coordinates are:
x = 4
y = 3
z = 0
>> spherical2rect
Enter vector length r: 5
Enter paln view angle theta in degrees: 0
Enter elevation angle phi in degrees: 36.87
The rectangular coordinates are:
x = 4
y = 0
z = 3
```

2.16 *(a)* A program that accepts a 3D vector in rectangular coordinates and calculates the vector in spherical coordinates is shown below:

```
%  Script file: rect2spherical.m
%
%  Purpose:
%    To calculate the spherical coordinates of a vector given
%    the 3D rectangular coordinates.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
% 1. 06/03/11    S. J. Chapman        Modified to use cart2sph
%
% Define variables:
%   x, y, z   -- Rectangular coordinates of vector
%   r         -- Length of vector
%   theta     -- Direction of vector (x,y) plane, in degrees
%   phi       -- Elevation angle of vector, in degrees

% Prompt the user for the input points
x = input('Enter x: ');
```

```
y = input('Enter y: ');
z = input('Enter z: ');

% Calculate polar coordinates.  Note that "180/pi" converts
% from radians to degrees.
[theta,phi,r] = cart2sph(x,y,z);
theta = theta * 180/pi;
phi = phi * 180/pi;

% Tell user
disp ('The spherical coordinates are:');
disp (['r     = ' num2str(r)]);
disp (['theta = ' num2str(theta)]);
disp (['phi   = ' num2str(phi)]);
```

When this program is executed, the results are:

```
>> rect2spherical
Enter x: 4
Enter y: 3
Enter z: 0
The spherical coordinates are:
r     = 5
theta = 36.8699
phi   = 0

>> rect2spherical
Enter x: 4
Enter y: 0
Enter z: 3
The spherical coordinates are:
r     = 5
theta = 0
phi   = 36.8699
```

(b) A program that accepts a 3D vector in spherical coordinates (with the angles θ and ϕ in degrees) and calculates the vector in rectangular coordinates is shown below:

```
%  Script file: spherical2rect.m
%
%  Purpose:
%    To calculate the 3D rectangular coordinates of a vector
%    given the spherical coordinates.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
% 1. 06/03/11    S. J. Chapman        Modified to use sph2cart
%
% Define variables:
%   x, y, z   -- Rectangular coordinates of vector
%   r         -- Length of vector
```

```
%   theta     -- Direction of vector (x,y) plane, in degrees
%   phi       -- Elevation angle of vector, in degrees

% Prompt the user for the input points
r = input('Enter vector length r: ');
theta = input('Enter plan view angle theta in degrees: ');
phi = input('Enter elevation angle phi in degrees: ');

% Calculate spherical coordinates.  Note that "pi/180" converts
% from radians to degrees.
[x,y,z] = sph2cart(theta*pi/180,phi*pi/180,r);

% Tell user
disp ('The 3D rectangular coordinates are:');
disp (['x = ' num2str(x)]);
disp (['y = ' num2str(y)]);
disp (['z = ' num2str(z)]);
```

When this program is executed, the results are:

```
>> spherical2rect
Enter vector length r: 5
Enter paln view angle theta in degrees: 36.87
Enter elevation angle phi in degrees: 0
The rectangular coordinates are:
x = 4
y = 3
z = 0
>> spherical2rect
Enter vector length r: 5
Enter paln view angle theta in degrees: 0
Enter elevation angle phi in degrees: 36.87
The rectangular coordinates are:
x = 4
y = 0
z = 3
```

2.17 A program to calculate the angle between two 2D vectors is shown below:

```
%  Script file: angle_between_vectors.m
%
%  Purpose:
%    To calculate the angle bewteen two 2D vectors.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/03/11    S. J. Chapman        Original code
%
% Define variables:
%   ux, uy    -- Components of vector 1
%   vx, vy    -- Components of vector 2
%   abs_u     -- Magnitude of u
%   abs_v     -- Magnitude of v
```

```
%   costheta  -- Cosine of angle bewteen the vectors
%   theta     -- Angle between the vectors (deg)

% Prompt the user for the input points
ux = input('Enter vector 1 x: ');
uy = input('Enter vector 1 y: ');
vx = input('Enter vector 2 x: ');
vy = input('Enter vector 2 y: ');

% Calculate the cosine of the angle between the vectors
abs_u = sqrt(ux^2 + uy^2);
abs_v = sqrt(vx^2 + vy^2);
costheta = (ux * vx + uy * vy) / (abs_u * abs_v);
theta = acos(costheta) * 180/pi;

% Tell user
disp (['The angle between the two vectors is ' num2str(theta) ' deg']);
```

When this program is executed, the results are:

```
>> angle_between_vectors
Enter vector 1 x: 1
Enter vector 1 y: 0
Enter vector 2 x: 1
Enter vector 2 y: 1
The angle between the two vectors is 45 deg
>> angle_between_vectors
Enter vector 1 x: 1
Enter vector 1 y: 1
Enter vector 2 x: -1
Enter vector 2 y: -1
The angle between the two vectors is 180 deg
```

2.18 A program to calculate the angle between two 3D vectors is shown below:

```
%  Script file: angle_between_vectors.m
%
%  Purpose:
%    To calculate the angle bewteen two 3D vectors.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/04/11    S. J. Chapman        Original code
%
% Define variables:
%   ux, uy, uz -- Components of vector 1
%   vx, vy, uz -- Components of vector 2
%   abs_u      -- Magnitude of u
%   abs_v      -- Magnitude of v
%   costheta   -- Cosine of angle bewteen the vectors
%   theta      -- Angle between the vectors (deg)

% Prompt the user for the input points
```

```
ux = input('Enter vector 1 x: ');
uy = input('Enter vector 1 y: ');
uz = input('Enter vector 1 z: ');
vx = input('Enter vector 2 x: ');
vy = input('Enter vector 2 y: ');
vz = input('Enter vector 2 z: ');

% Calculate the cosine of the angle between the vectors
abs_u = sqrt(ux^2 + uy^2 + uz^2);
abs_v = sqrt(vx^2 + vy^2 + vz^2);
costheta = (ux * vx + uy * vy + uz * vz) / (abs_u * abs_v);
theta = acos(costheta) * 180/pi;

% Tell user
disp (['The angle between the two vectors is ' num2str(theta) ' deg']);
```

When this program is executed, the results are:

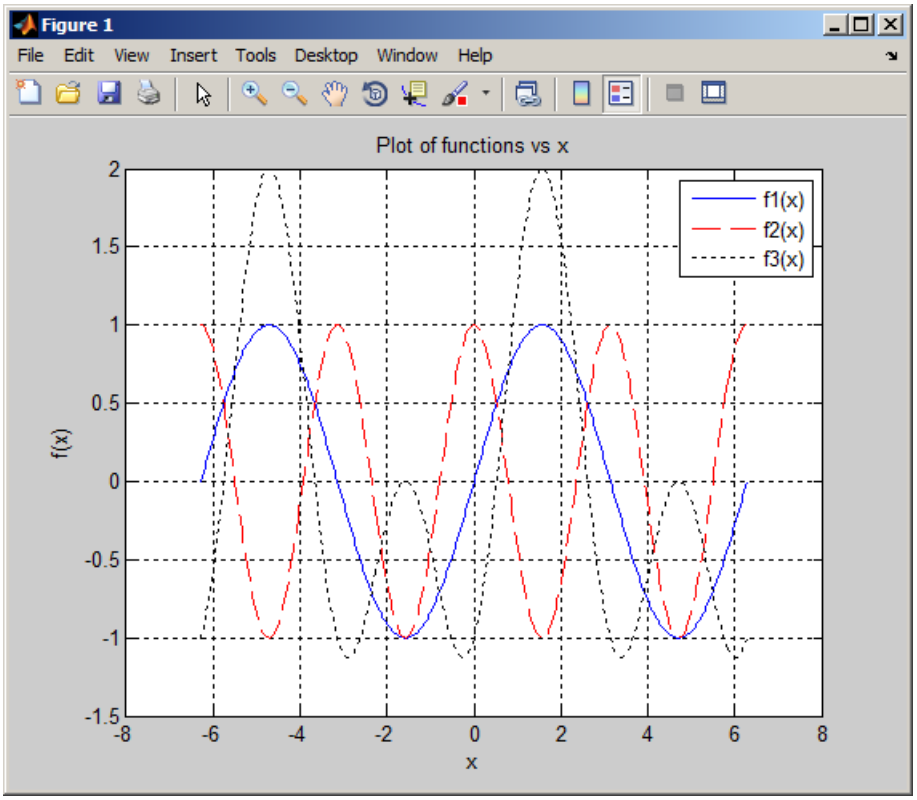
```
>> angle_between_vectors
Enter vector 1 x: 1
Enter vector 1 y: 0
Enter vector 1 z: 0
Enter vector 2 x: 0
Enter vector 2 y: 0
Enter vector 2 z: 1
The angle between the two vectors is 90 deg
```

2.19 In Chapter 2, we have learned how to plot more than one curve on a figure by including all of them in a single `plot` statement. In a later chapter we will learn how to plot the curves one at a time on top of each other. For now, though, we will calculate all three functions and plot them at the same time.

The code to create this plot is shown below.

```
x = -2*pi:pi/100:2*pi;
f1 = sin(x);
f2 = cos(2*x);
f3 = f1 - f2;
plot(x,f1,'b-',x,f2,'r--',x,f3,'k:');
title('Plot of functions vs x');
xlabel('x');
ylabel('f(x)');
legend('f1(x)','f2(x)','f3(x)');
grid on;
```

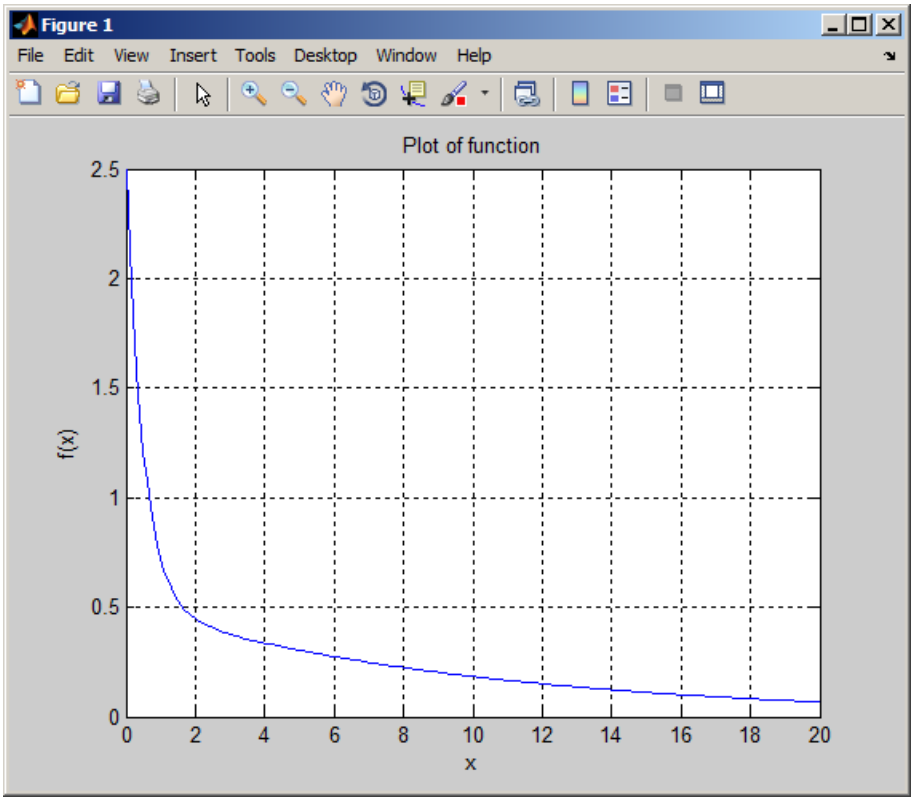
When this program is executed, the results are:



2.20 The code to create a linear is shown below.

```
x = 0:0.1:20;
y = 2 * exp(-2*x) + 0.5 * exp(-0.1*x);
plot(x,y);
title('Plot of function');
xlabel('x');
ylabel('f(x)');
grid on;
```

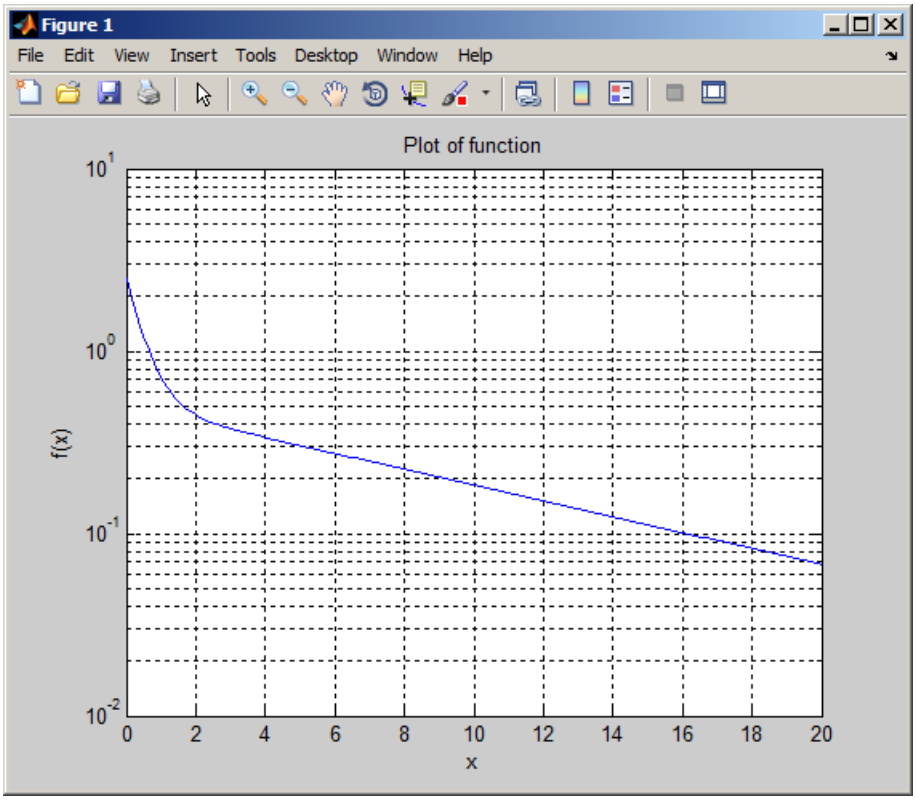
When this program is executed, the results are:



The code to create a semilog plot is shown below.

```
x = 0:0.1:20;
y = 2 * exp(-2*x) + 0.5 * exp(-0.1*x);
semilogy(x,y);
title('Plot of function');
xlabel('x');
ylabel('f(x)');
grid on;
```

When this program is executed, the results are:



Note that the exponential functions appear as straight line segments on the semilog plot.

2.21 The angular acceleration of the shaft is given by:

$$\boldsymbol{\tau} = I\boldsymbol{\alpha}$$

So solving for α

$$\boldsymbol{\alpha} = \frac{\boldsymbol{\tau}}{I}$$

The angular acceleration is thus

```
>> tau = 20;
>> I = 15;
>> alpha = tau / I
alpha =
    1.3333
```

The angular acceleration is $\alpha = 1.333$ rad/s

2.22 A program to calculate power in dBm is shown below:

```
%  Script file: decibel.m
%
%  Purpose:
%    To calculate the dBm corresponding to a user-supplied
%    power in watts.
```

```
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/04/11    S. J. Chapman        Original code
%
% Define variables:
%   dBm       -- Power in dBm
%   pin       -- Power in watts

% Prompt the user for the input power.
pin = input('Enter the power in watts: ');

% Calculate dBm
dBm = 10 * log10( pin / 1.0e-3 );

% Tell user
disp (['Power = ' num2str(dBm) ' dBm']);
```

When this program is executed, the results are:

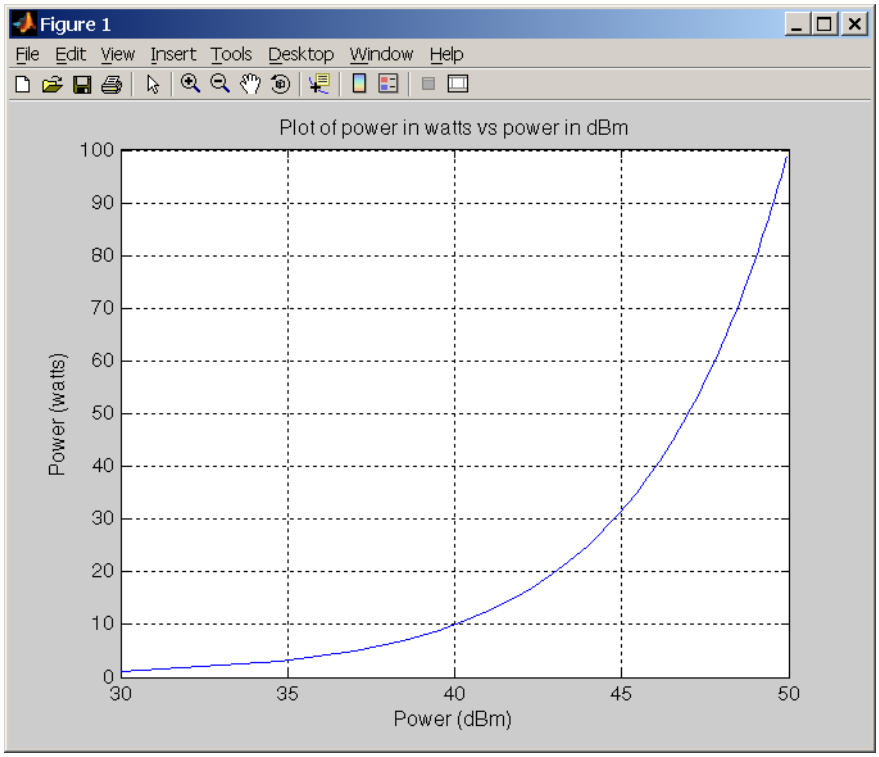
```
» decibel
Enter the power in watts: 10
Power = 40 dBm
» decibel
Enter the power in watts: 0.1
Power = 20 dBm
```

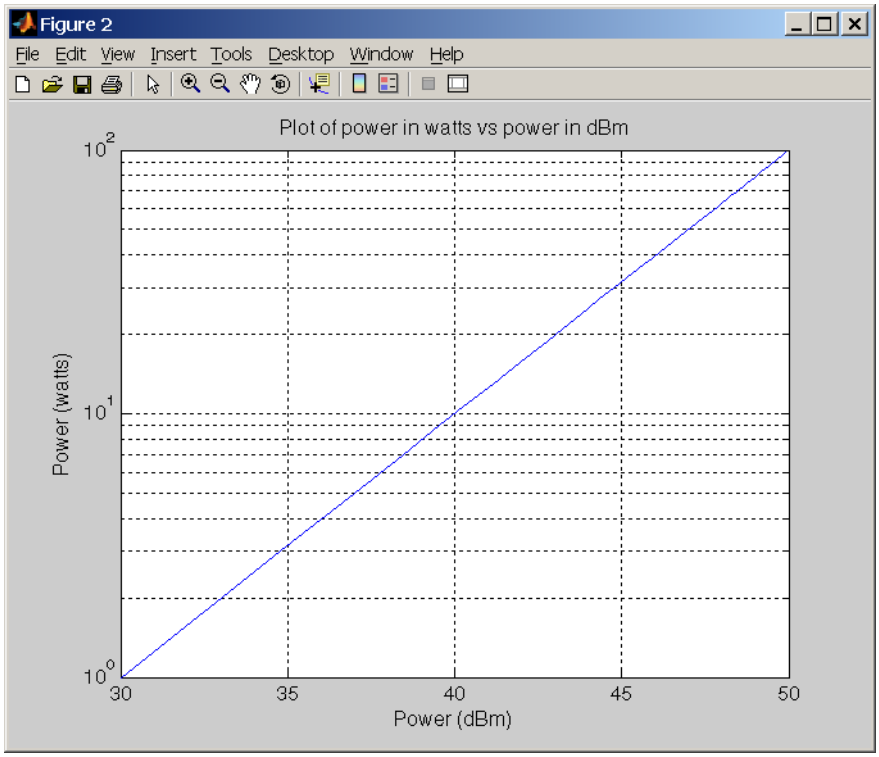
When this program is executed, the results are:

```
%  Script file: db_plot.m
%
%  Purpose:
%    To plot power in watts vs power in dBm on a linear and
%    log scale.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/04/11    S. J. Chapman        Original code
%
% Define variables:
%   dBm       -- Power in dBm
%   pin       -- Power in watts

% Create array of power in watts
pin = 1:2:100;

% Calculate power in dBm
dBm = 10 * log10( pin / 1.0e-3 );

% Plot on linear scale
figure(1);
plot(dBm,pin);
```

```
title('Plot of power in watts vs power in dBm');
xlabel('Power (dBm)');
ylabel('Power (watts)');
grid on;

% Plot on semilog scale
figure(2);
semilogy(dBm,pin);
title('Plot of power in watts vs power in dBm');
xlabel('Power (dBm)');
ylabel('Power (watts)');
grid on;
```

When this program is executed, the results are:



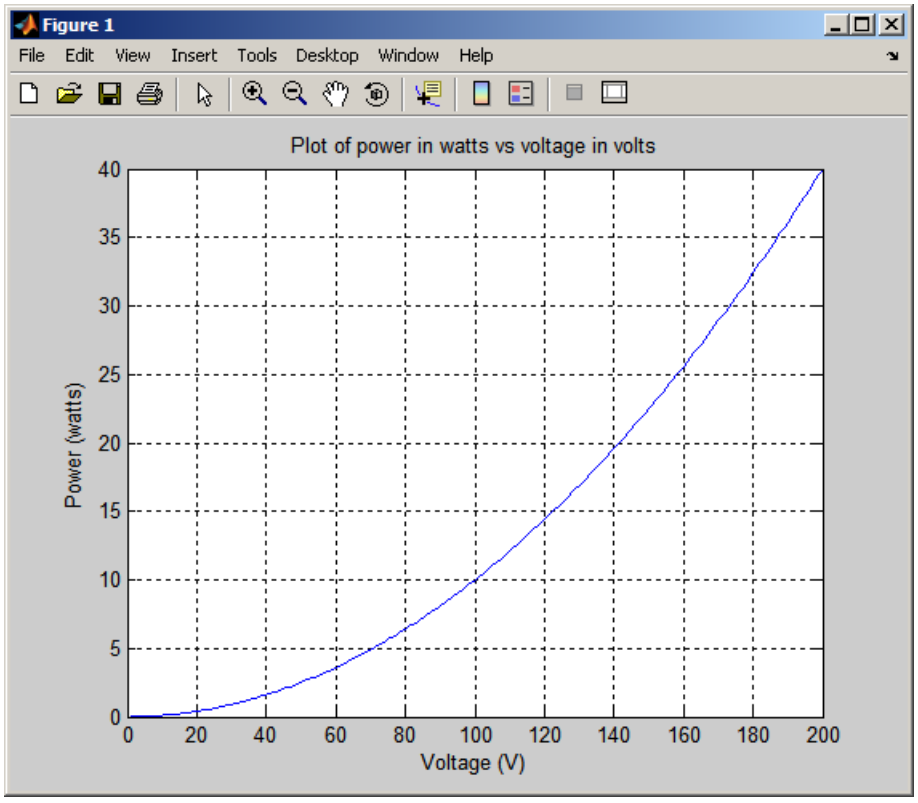


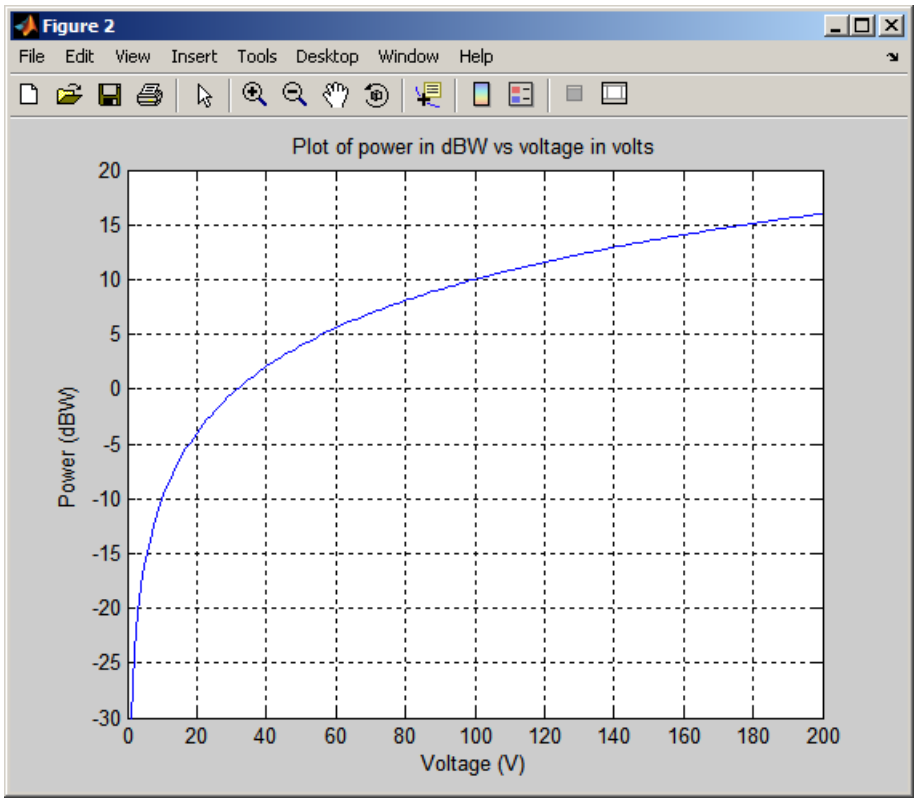
2.23 A program to calculate and plot the power consumed by a resistor as the voltage across the resistor is varied from 1 to 200 volts shown below:

```
%  Script file: p_resistor.m
%
%  Purpose:
%    To plot the power consumed by a resistor as a function
%    of the voltage across the resistor on both a linear and
%    a log scale.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/04/11    S. J. Chapman        Original code
%
% Define variables:
%   ir    -- Current in the resistor (A)
%   pr    -- Power in the resistor (W)
%   r     -- Resistance of resistor (ohms)
%   vr    -- Voltage across the resistor (V)
%   vr_db -- Voltage across the resistor (dBW)

% Set the resistance
r = 1000;

% Create array of voltage across the resistor
vr = 1:200;

% Calculate the current flow through the resistor
```

```
ir = vr / r;

% Calculate the power consumed by the resistor in watts
pr = ir .* vr;

% Calculate the power consumed by the resistor in dBW
pr_db = 10 * log10(pr);

% Plot on linear scale
figure(1);
plot(vr,pr);
title('Plot of power in watts vs voltage in volts');
xlabel('Voltage (V)');
ylabel('Power (watts)');
grid on;

% Plot on semilog scale
figure(2);
plot(vr,pr_db);
title('Plot of power in dBW vs voltage in volts');
xlabel('Voltage (V)');
ylabel('Power (dBW)');
grid on;
```

The resulting plots are shown below.





2.24 A program to calculate cosh(x) both from the definition and using the MATLAB intrinsic function is shown below. Note that we are using `fprintf` to display the results, so that we can control the number of digits displayed after the decimal point:

```
%  Script file: cosh1.m
%
%  Purpose:
%    To calculate the hyperbolic cosine of x.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/04/11    S. J. Chapman        Original code
%
% Define variables:
%   x          -- Input value
%   res1       -- cosh(x) from the definition
%   res2       -- cosh(x) from the MATLAB function

% Prompt the user for the input power.
x = input('Enter x: ');

% Calculate cosh(x)
res1 = ( exp(x) + exp(-x) ) / 2;
res2 = cosh(x);

% Tell user
fprintf('Result from definition = %14.10f\n',res1);
```

```
fprintf('Result from function   = %14.10f\n',res2);
```

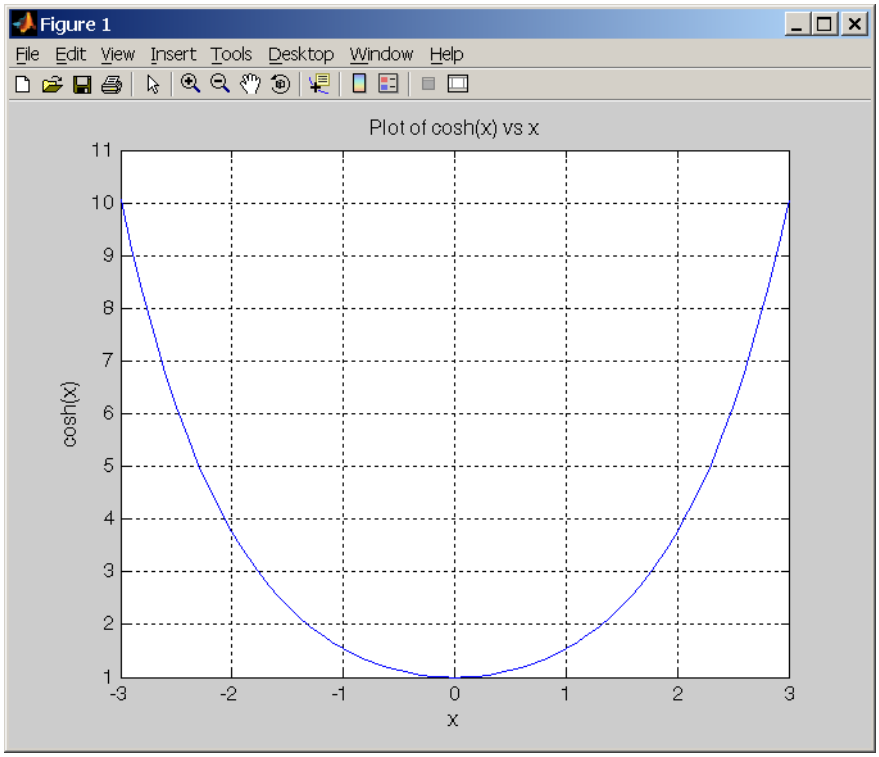
When this program is executed, the results are:

```
» cosh1
Enter x: 3
Result from definition =  10.0676619958
Result from function   =  10.0676619958
```

A program to plot cosh *x* is shown below:

```
%  Script file: cosh_plot.m
%
%  Purpose:
%    To plot cosh x vs x.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/04/11    S. J. Chapman        Original code
%
% Define variables:
%   x         -- input values
%   coshx     -- cosh(x)

% Create array of power in input values
x = -3:0.1:3;

% Calculate cosh(x)
coshx = cosh(x);

% Plot on linear scale
plot(x,coshx);
title('Plot of cosh(x) vs x');
xlabel('x');
ylabel('cosh(x)');
grid on;
```

The resulting plot is shown below. Note that the function reaches a minimum value of 1.0 at $x = 0$.



2.25 A program to calculate the energy stored in a spring is shown below:

```
%  Script file: spring.m
%
%  Purpose:
%    To calculate the energy stored in a spring.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/04/11    S. J. Chapman        Original code
%
% Define variables:
%   energy    -- Stored energy (J)
%   f         -- Force on spring (N)
%   k         -- Spring constant (N/m)
%   x         -- Displacement (m)

% Prompt the user for the input force and spring constant.
f = input('Enter force on spring (N): ');
k = input('Enter spring constant (N/m): ');

% Calculate displacement x
x = f/k;

% Calculate stored energy
energy = 0.5 * k * x^2;
```

```
% Tell user
fprintf('Displacement = %.3f meters\n',x);
fprintf('Stored energy = %.3f joules\n',energy);
```

When this program is executed, the results are as shown below. The second spring stores the most energy.

```
» spring
Enter force on spring (N): 20
Enter spring constant (N/m): 200
Displacement = 0.100 meters
Stored energy = 1.000 joules
» spring
Enter force on spring (N): 30
Enter spring constant (N/m): 250
Displacement = 0.120 meters
Stored energy = 1.800 joules
» spring
Enter force on spring (N): 25
Enter spring constant (N/m): 300
Displacement = 0.083 meters
Stored energy = 1.042 joules
» spring
Enter force on spring (N): 20
Enter spring constant (N/m): 800
Displacement = 0.050 meters
Stored energy = 0.500 joules
```

2.26 A program to calculate the resonant frequency of a radio is shown below:

```
%  Script file: radio.m
%
%  Purpose:
%    To calculate the resonant frequency of a radio.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/06/11    S. J. Chapman        Original code
%
% Define variables:
%   c         -- Capacitance (F)
%   freq      -- Resonant frequency (Hz)
%   l         -- Inductance (H)

% Prompt the user for the input force and spring constant.
l = input('Enter inductance in henrys: ');
c = input('Enter capacitance in farads: ');

% Calculate resonant frequency
freq = 1 / ( 2 * pi * sqrt(l*c) );

% Tell user
```

```
fprintf('Resonant frequency = %.1f Hz\n',freq);
```

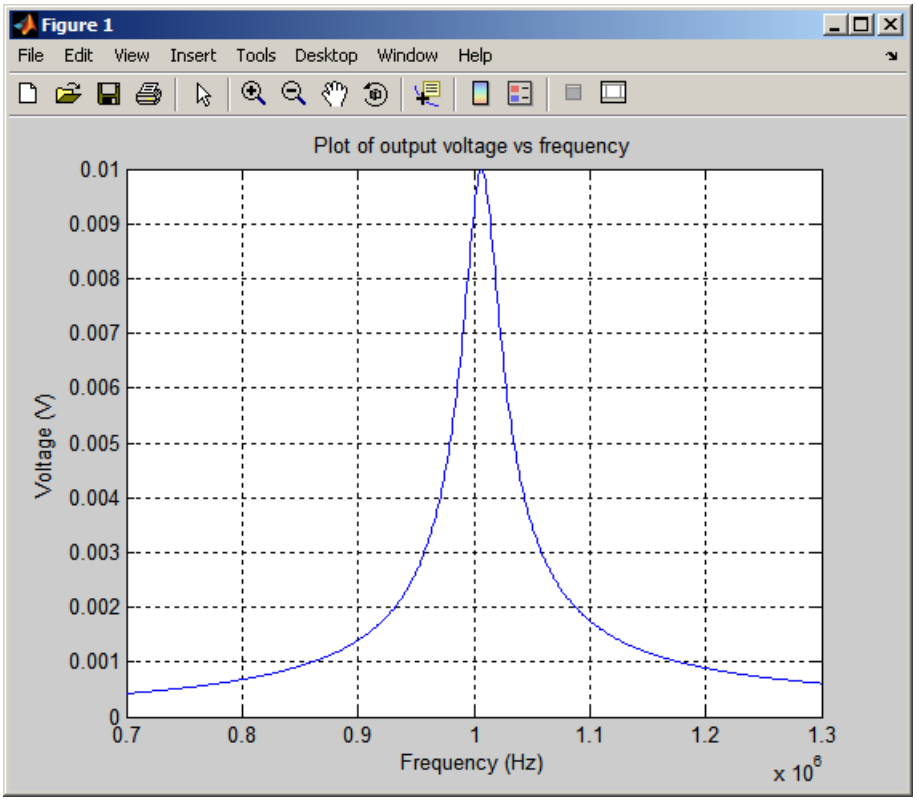
When this program is executed, the results are:

```
» radio
Enter inductance in henrys: 0.25e-3
Enter capacitance in farads: 0.1e-9
Resonant frequency = 1006584.2 Hz
```

2.27 *(a)* A program to calculate the frequency response of a radio receiver is shown below:

```
%  Script file: radio2.m
%
%  Purpose:
%    To plot the frequency response of a radio receiver.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/06/11    S. J. Chapman        Original code
%
% Define variables:
%   c         -- Capacitance (F)
%   freq      -- Resonant frequency (Hz)
%   l         -- Inductance (H)
%   r         -- resistance (ohms)
%   v         -- output viltage (V)
%   v0        -- input voltage (V)
%   w         -- Angular frequency (rad/s)

% Initialise values
c  = 0.1e-9;
l  = 0.25e-3;
r  = 50;
v0 = 10e-3;

% Create an array of frequencies centered on 1 MHz,
% which is the resonant frequency
freq = (0.7:0.001:1.3) * 1e6;

% Calculate w
w = 2 * pi * freq;

% Calculate output voltage
v =  v0 .* r ./ sqrt( r^2 + (w.*l - 1./(w.*c)).^2 );

% Plot on linear scale
plot(freq,v);
title('Plot of output voltage vs frequency');
xlabel('Frequency (Hz)');
ylabel('Voltage (V)');
grid on;
```

The resulting frequency response is shown below. Note that the function reaches a minimum value of 1.0 at $x = 0$.



(b) The resonant frequency of this circuit is about 1.007 MHz. If the frequency is changed to 1.1 MHz, the output voltage will be 1.75 mV instead of the 10 mV at the resonant frequency. This receiver is not very selective—real radios do *much* better.

(c) The output voltage drops from 10 mV to 5 mV at 0.979 MHz and 1.035 MHz.

2.28 A program to calculate the output power of the receiver for a given input voltage and frequency is shown below:

```
%  Script file: radio3.m
%
%  Purpose:
%    To calculate the output power of a radio receiver.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/06/11    S. J. Chapman        Original code
%
% Define variables:
%   c          -- Capacitance (F)
%   freq       -- Resonant frequency (Hz)
%   l          -- Inductance (H)
%   r          -- resistance (ohms)
%   p          -- output power (W)
%   v          -- output viltage (V)
```

```
%   v0        -- input voltage (V)
%   w         -- Angular frequency (rad/s)

% Initialise values
c  = 0.1e-9;
l  = 0.25e-3;
r  = 50;

% Get voltage and frequency
v0 = input('Enter voltage (V): ');
freq = input('Enter frequency  (Hz): ');

% Calculate w
w = 2 * pi * freq;

% Calculate output voltage
v =  v0 .* r ./ sqrt( r^2 + (w.*l - 1./(w.*c)).^2 );

% Calculate output power (=v^2/r)
p = v^2 / r;

% Tell user
fprintf('Output power = %f W\n',p);
```

When this program is executed, the results are:

```
» radio3
Enter voltage (V): 1
Enter frequency  (Hz): 1e6
Output power = 0.017061 W
» radio3
Enter voltage (V): 1
Enter frequency  (Hz): 0.95e6
Output power = 0.001388 W
```

The power ration in dB is

```
» dB = 10*log10(0.017061/0.001388)
dB =
   10.8962
```

The second signal is *suppressed* by about 11 dB compared to the first signal.

2.29 *(a)* The solution to the equations is:

```
>> A = [ 2 2 3; 4 5 6; 7 8 9];
>> b = [1; 2; 3];
>> x = A \ b
x =
    0.0000
         0
    0.3333
```

(b) The solution to the equations is:

```
>> A = [ 1 2 3; 4 5 6; 7 8 9];
>> b = [1; 2; 3];
>> x = A \ b

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.541976e-018.
x =
   -0.3333
    0.6667
         0
```

This set of equations is singular.

(c) The solution to the equations is:

```
>> A = [-2  5  1  3  4 -1  2 -1 -5 -2;
       6  4 -1  6 -4 -5  3 -1  4  2;
      -6 -5 -2 -2 -3  6  4  2 -6  4;
       2  4  4  4  5 -4  0  0 -4  6;
      -4 -1  3 -3 -4 -4 -4  4  3 -3;
       4  3  5  1  1  1  0  3  3  6;
       1  2 -2  0  3 -5  5  0  1 -4;
      -3 -4  2 -1 -2  5 -1 -1 -4  1;
       5  5 -2 -5  1  4 -1  0 -2 -3;
      -5 -2 -5  2  1 -3  4 -1 -4  4];
>> b = [-5; -6; -7; 0; 5; -8; 1; -4; -7; 6]
>> x = A \ b
x =
   -0.0502
   -1.2196
   -1.1074
    0.2344
    1.0318
   -1.0158
   -1.4402
    0.3531
    0.1197
   -0.0454
```

2.30 *(a)* A program for calculating the turning radius of the aircraft is shown below:

```
%  Script file: turning.m
%
%  Purpose:
%    To calculate the turning radius of an aircraft flying
%    in a circle, based on speed and max g.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/06/11    S. J. Chapman        Original code
%
% Define variables:
```

```
%   g         -- Max acceleration (g)
%   grav      -- Acceleration of gravity (9.81 m/s2)
%   mach1     -- Speed of sound (340 m/s)
%   radius    -- Turning radius (m)
%   speed     -- Aircraft speed in Mach

% Initialise values
grav  = 9.81;
mach1 = 340;

% Get speed and max g
speed = input('Enter speed (Mach): ');
g = input('Enter max acceleration (g): ');

% Calculate radius
radius = (speed * mach1).^ 2 / ( g * grav );

% Tell user
fprintf('Turning radius = %f m\n',radius);
```

When this program is executed, the results are:

```
>> turning
Enter speed (Mach): .85
Enter max acceleration (g): 2
Turning radius = 4256.931702 m
```

The turning radius is 4257 meters.

(b) When this program is executed with the new speed, the results are:

```
>> turning
Enter speed (Mach): 1.5
Enter max acceleration (g): 2
Turning radius = 13256.880734 m
```

The turning radius is now 13257 meters.

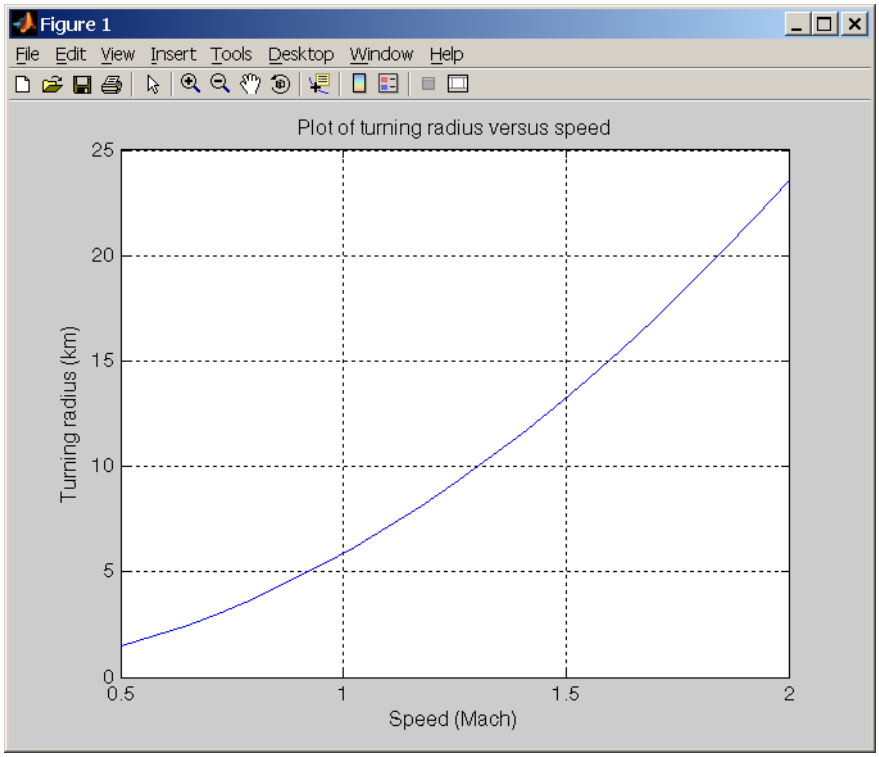
(c) A program to plot the turning radius as a function of speed for a constant max acceleration is shown below:

```
%  Script file: turning2.m
%
%  Purpose:
%    To plot the turning radius of an aircraft as a function
%    of speed.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/06/11    S. J. Chapman        Original code
%
% Define variables:
%   g         -- Max acceleration (g)
```

```
%   grav      -- Acceleration of gravity (9.81 m/s2)
%   mach1     -- Speed of sound (340 m/s)
%   max_speed -- Maximum speed in Mach numbers
%   min_speed -- Minimum speed in Mach numbers
%   radius    -- Turning radius (m)
%   speed     -- Aircraft speed in Mach

% Initialise values
grav  = 9.81;
mach1 = 340;

% Get speed and max g
min_speed = input('Enter min speed (Mach): ');
max_speed = input('Enter min speed (Mach): ');
g = input('Enter max acceleration (g): ');

% Calculate range of speeds
speed = min_speed:(max_speed-min_speed)/20:max_speed;

% Calculate radius
radius = (speed * mach1).^ 2 / ( g * grav );

% Plot the turning radius versus speed
plot(speed,radius/1000);
title('Plot of turning radius versus speed');
xlabel('Speed (Mach)');
ylabel('Turning radius (km)');
grid on;
```

When this program is executed, the results are as shown below:

```
>> turning2
Enter min speed (Mach): 0.5
Enter min speed (Mach): 2.0
Enter max acceleration (g): 2
```



(d) When this program is executed, the results are:

```
>> turning
Enter speed (Mach): 1.5
Enter max acceleration (g): 7
Turning radius = 3787.680210 m
```

The turning radius is now 3788 meters.

(e) A program to plot the turning radius as a function of centripetal acceleration is shown below:

```
%  Script file: turning3.m
%
%  Purpose:
%    To plot the turning radius of an aircraft as a function
%    of centripetal acceleration.
%
%  Record of revisions:
%      Date       Programmer          Description of change
%      ====       ==========          =====================
%    06/06/11    S. J. Chapman        Original code
%
% Define variables:
%   g          -- Acceleration (g)
%   grav       -- Acceleration of gravity (9.81 m/s2)
%   mach1      -- Speed of sound (340 m/s)
%   max_g      -- Maximum accleration in g's
%   min_g      -- Minimum accleration in g's
%   radius     -- Turning radius (m)
```

```
%   speed     -- Aircraft speed in Mach

% Initialise values
grav  = 9.81;
mach1 = 340;

% Get speed and max g
speed = input('Enter speed (Mach): ');
min_g = input('Enter min acceleration (g): ');
max_g = input('Enter min acceleration (g): ');

% Calculate range of accelerations
g = min_g:(max_g-min_g)/20:max_g;

% Calculate radius
radius = (speed * mach1).^ 2 ./ ( g * grav );

% Plot the turning radius versus speed
plot(g,radius/1000);
title('Plot of turning radius versus acceleration');
xlabel('Centripetal acceleration (g)');
ylabel('Turning radius (km)');
grid on;
```

When this program is executed, the results are as shown below:

```
>> turning3
Enter speed (Mach): 0.85
Enter min acceleration (g): 2
Enter min acceleration (g): 8
```